## Package ‘mvMORPH’

March 17, 2015

**Type**  Package  
**Title**  Multivariate Comparative Tools for Fitting Evolutionary Models to Morphometric Data  
**Version**  1.0.3  
**Date**  2015-03-17  
**Author**  Julien Clavel, with contributions from Aaron King, and Emmanuel Paradis  
**Maintainer**  Julien Clavel &lt;julien.clavel@hotmail.fr&gt;  
**Description**  Fits multivariate (Brownian Motion, Early Burst, ACDC, Ornstein-Uhlenbeck and Shifts) models of continuous traits evolution on trees.  
**Depends**  R(&gt;= 2.9.1), phytools, ape, corpcor, subplex, spam  
**License**  GPL (&gt;= 2.0)  
**NeedsCompilation**  yes  
**Repository**  CRAN  
**Date/Publication**  2015-03-17 20:59:58  

### R topics documented:

- mvMORPH-package ........................................... 2  
- LRT .......................................................... 2  
- mv.Precalc .................................................. 4  
- mvBM ......................................................... 6  
- mvEB ......................................................... 10  
- mvLL .......................................................... 13  
- mvOU ......................................................... 16  
- mvSHIFT ...................................................... 21  
- mvSIM ......................................................... 25  

**Index** 28
mvMORPH-package

Multivariate Comparative Methods for Fitting Evolutionary Models to Morphometric Data

Description

Fits of multivariate evolutionary models (with one or multiple selective regimes) dedicated to morphometrics or biometric continuous data with covariation. Testing for a phylogenetic signal in a multivariate dataset, changes in rate or mode of evolution of continuous traits, simulating multivariate traits evolution, computing the likelihood of multivariate models and other things...

Details

Package: mvMORPH
Type: Package
Version: 1.0.3
Date: 2013-07-22
License: GPL (>=2.0)

Author(s)

Julien Clavel
Maintainer: Julien Clavel <julien.clavel@hotmail.fr>

References

Clavel et al. (submitted to Methods in Ecology and Evolution)

See Also

mvOU mvBM mvEB mvSHIFT mvSIM mvLL LRT brownie.lite evol.vcv

LRT

Likelihood Ratio Test

Description

This function compares the fit of two nested models of trait evolution with a loglikelihood-ratio statistic.
**Usage**

LRT(model1, model2, echo = TRUE)

**Arguments**

- **model1**: The most parameterized model. A fitted object from an mvMORPH model.
- **model2**: The second model under comparison (fitted object).
- **echo**: Whether to return the result or not.

**Details**

The LRT function extracts the log-likelihood of two nested models to compute the loglikelihood-ratio statistic which is compared to a Chi-square distribution. Note that if the models are not nested, the LRT is not an appropriate test and you should rely instead on Information criteria and evidence ratios.

**Value**

- **pval**: The p-value of the LRT test (comparison with Chi-square distribution).
- **ratio**: The LRT (Loglikelihood-ratio test) statistic.
- **ddf**: The number of degrees of freedom between the two models.
- **model1**: Name of the first model.
- **model2**: Name of the second model.

**Author(s)**

Julien Clavel

**References**


**See Also**

mvMORPH mvOU mvEB mvBM mvSHIFT

**Examples**

```r
## Simulated dataset
set.seed(14)
# Generating a random tree
tree<--pbtree(n=50)

# Setting the regime states of tip species
sta<as.vector(c(rep("Forest",20),rep("Savannah",30))); names(sta)<tree$tip.label

# Making the simmap tree with mapped states
```
mv.Precalc

Model parameterization for the various mvMORPH functions

Description

This function allows computing the fixed parameters or objects needed by the mvMORPH functions. This could be useful in bootstrap-like computations (see exemple)

Usage

mv.Precalc(tree, nb.traits = 1, scale.height = FALSE, param = list(pivot = "MMD", method = c("sparse"), smean = TRUE, model = "OUM"))

Arguments

tree A "phylo" (or SIMMAP like) object representing the tree for which we want to precalculate parameters.
nb.traits The number of traits involved in the subsequent analysis.
scale.height Whether the tree should be scaled to unit length.
param A list of parameters used in the computations (see details)

Details

The mv.Precalc function allows the pre-computation of the fixed parameters required by the different mvMORPH models (e.g., the design matrix, the vcv matrix, the sparsity structure...). In the "param" list you should provide the details about the model fit:

- model name (e.g., "OUM", "OU1")
mv.Precalc

- `method` (which kind of algorithm is used for computing the log-likelihood).
- `smean` (whether there is one ancestral state per traits or per selective regimes - for mvBM only).

Additional parameters can be fixed: `root` (estimation of the ancestral state for the Ornstein-Uhlenbeck model, see `?mvOU`). `pivot` (pivot method used by the "sparse" matrix method for computing the log-likelihood, see `?spam`).

**Value**

An object of class "mvmorph.precalc" which can be used in the "precalc" argument of the various mvMORPH functions.

**Note**

This function is mainly used internally; it is still in development. A misuse of this function can result in a crash of the R session.

**Author(s)**

Julien Clavel

**See Also**

mvMORPH mvOU mvEB mvBM mvSHIFT mvLL

**Examples**

```r
set.seed(14)
# Generating a random tree
tree<-pbtree(n=50)

# Simulate two correlated traits evolving along the phylogeny according to a
# Ornstein-Uhlenbeck process
alpha<-matrix(c(2.1,1,1.3),2,2)
sigma<-matrix(c(1,0.5,0.5,0.8),2,2)
theta<-c(3,1)
nsim<-50
simul<-mvSIM(tree,param=list(sigma=sigma, alpha=alpha, ntraits=2, mu=theta, names_traits=c("head.size","mouth.size")), model="OU1", nsim=nsim)

# Do the pre-calculations
precal<-mv.Precalc(tree, nb.traits=2, param=list(method="sparse", model="OU1", root=FALSE))

mvOU(tree, simul[[1]], method="sparse", model="OU1", precalc=precal, param=list(decomp="symmetric"))

### Bootstrap
# Fit the model to the "nsim" simulated datasets
results<-lapply(1:nsim,function(x){
  mvOU(tree, simul[[x]], method="sparse", model="OU1", precalc=precal, #
    param=list(decomp="symmetric"))
  }
)
mvBM

Multivariate Brownian motion models of continuous traits evolution

Description

This function allows the fitting of multivariate multiple rates of evolution under a Brownian motion model. This function can also fit constrained models.

Usage

mvBM(tree, data, error = NULL, model = c("BMM", "BM1"), param =
list(constraint = FALSE, smean = TRUE), method = c("rpf", "pic",
"sparse", "inverse", "pseudoinverse"), scale.height = FALSE,
optimization = c("L-BFGS-B", "Nelder-Mead", "subplex"),
control = list(maxit = 20000), precalc = NULL, diagnostic = TRUE,
echo = TRUE)

Arguments

tree
Phylogenetic tree in SIMMAP format by default. A "phylo" object can also be used with the "BM1" model.
data
Matrix or data frame with species in rows and continuous traits in columns (preferentially with names and in the same order than in the tree).
error
Matrix or data frame with species in rows and continuous traits standard error (squared) in columns.
model
"BMM" for a multi-rate and multi-selective regimes, and "BM1" for a unique rate of evolution per trait.
param
List of arguments to be passed to the function. See details.
method
Choose between "rpf", "sparse", "inverse", "pseudoinverse", or "pic" for log-likelihood computation during the fitting process. See details.
scale.height
Whether the tree should be scaled to length 1 or not.
optimization  Methods used by the optimization routines (see ?optim and ?subplex for details).
control     Max. bound for the number of iteration of the optimizer; other options can be fixed on the list (see ?optim or ?subplex).
precalc     Optional. precalculation of fixed parameters. See ?mvvmorph.Precalc.
diagnostic Whether the diagnostics of convergence should be returned or not.
echo        Whether the results must be returned or not.

Details

The mvBM function fits a multivariate Brownian Motion (BM) process, with unique or multiple BM rates (see O’Meara et al., 2006; Revell and Collar, 2009). Note that the function uses the non-censored approach of O’Meara et al. (2006) by default (i.e., a common ancestral state is assumed for the different regimes), but it is possible to specify multiple ancestral states (i.e., one for each regime) through the "smean" parameter (smean=FALSE) in the "param" list.

The "method" argument allows the user to try different algorithms for computing the log-likelihood. The "rpf" and "sparse" methods use fast GLS algorithms based on factorization for avoiding the computation of the inverse of the variance-covariance matrix and its determinant involved in the log-likelihood estimation. The "inverse" approach uses the "stable" standard explicit computation of the inverse and determinant of the matrix and is therefore slower. The "pseudoinverse" method uses a generalized inverse that is safer for matrix near singularity but very time consuming. The "pic" method uses a very fast algorithm based on independent contrasts. It should be used with strictly dichotomic trees (i.e., no polytomies) and is currently not available for the multivariate "BMM" model. See ?mvLL for more details on these computational methods.

The "param" list arguments:

"constraint" - The "constraint" argument in the "param" list allows the user to compute the joint likelihood for each trait by assuming they evolved independently (constraint="diagonal"). If "constraint=TRUE", the model forces the sigma values for each studied trait to be the same using the constrained Cholesky decomposition proposed by Adams (2013), which allows considering the covariances between traits. This approach is extended here to the multi-rate case, by specifying that the rates must be the same in different parts of the tree (common selective regime). Significance of model fit can be assessed using the "LRT" function. See example below and ?LRT.

"smean" - Default set to TRUE. If FALSE, the ancestral state for each selective regime is estimated (e.g., Thomas et al., 2006).

"sigma" - Starting values for the likelihood estimation. By default the theoretical expected values are used as starting values for the likelihood optimization (for measurement errors, multiple rates,...). The user can specify starting values with a list() object for the "BMM" model (e.g., two objects in the list for a two-regime analysis), or a simple vector of values for the "BM1" model. The parameterization is done using a Cholesky factorization (Pinheiro & Bates, 1996). Thus, you should provide p*(p+1) values, with p the number of traits (e.g., random numbers or the values from the cholesky factor of a symmetric positive definite sigma matrix - see example below).

If no selective regimes are specified the function works only with the model "BM1".

N.B.: Mapping of ancestral states can be done using the "make.simmap", "make.era.map" or "paintSub-Tree" functions from the "phytools" package.
Value

- **LogLik**: The log-likelihood of the optimal model.
- **AIC**: Akaike Information Criterion for the optimal model.
- **AICc**: Sample size-corrected AIC.
- **theta**: Estimated ancestral states.
- **sigma**: Evolutionary rates matrix for each selective regime.
- **convergence**: Convergence status of the optimizing function; "0" indicates convergence (See ?optim for details).
- **hessian**: Hessian matrix of second order partial derivatives at the MLE (See ?mvOU for details).
- **hess.values**: Reliability of the likelihood estimates calculated through the eigen-decomposition of the hessian matrix. "0" means that a reliable estimate has been reached. (See ?mvOU).
- **param**: List of model fit parameters (optimization, method, model, number of parameters...).

Note

The "pic" method is not currently implemented for the multivariate "BMM" model.

Author(s)

- Julien Clavel

References


See Also

- mvMORPH
- mvOU
- mvEB
- mvSHIFT
- mvSIM
- LRT
- optim
- brownie.lite
- evol.vcv
- make.simmap
- make.era.map
- paintSubTree
**Examples**

```r
## Toy example

# Simulated dataset
set.seed(14)
# Generating a random tree
tree<-pbtree(n=50)

# Setting the regime states of tip species
sta<-as.vector(c(rep("Forest",20),rep("Savannah",30))); names(sta)<-tree$tip.label

# Making the simmap tree with mapped states
tree<-make.simmap(tree,sta, model="ER", nsim=1)
col<-c("blue","orange"); names(col)<-c("Forest","Savannah")

# Plot of the phylogeny for illustration
plotSimmap(tree,col,fsiz=0.6,node.numbers=FALSE,lwd=3, pts=FALSE)

# Simulate the traits
sigma<-matrix(c(0.1,0.05,0.05,0.1),2)
theta<0.0)
data<-mvSIM(tree, param=list(sigma=sigma, ntraits=2, mu=theta,
    names_traits=c("head.size","mouth.size")), model="BM1", nsim=1)

## Fitting the models

# BMM - Analysis with multiple rates
mvBM(tree, data)

# BM1 - Analysis with a unique rate matrix
fit1<-mvBM(tree, data, model="BM1", method="pic")

# BM1 constrained
fit2<-mvBM(tree, data, model="BM1", method="pic", param=list(constraint=TRUE))

# Comparison with LRT test
LRT(fit1,fit2)

# Random starting values
mvBM(tree, data, model="BMM", method="sparse", param=list(sigma=list(runif(3), runif(3)))

# Specified starting values (from the Cholesky factor)
chol_factor<-chol(sigma)
starting_values<-chol_factor[upper.tri(chol_factor,TRUE)]
mvBM(tree, data, model="BMM", method="sparse",
    param=list(sigma=list(starting_values, starting_values)))

# Multiple mean
mvBM(tree, data, model="BMM", method="sparse", param=list(smean=FALSE))

## FAST FOR THE UNIVARIATE CASE!!
# set.seed(14)
```
mvEB

Multivariate Early Burst model of continuous traits evolution

Description

This function fits multivariate Early Burst (EB) or ACDC models of evolution.

Usage

mvEB(tree, data, error = NULL, param = list(low = -3, up = 0), method = c("rpf", "sparse", "inverse", "pseudoinverse", "pic"), scale.height = FALSE, optimization = c("L-BFGS-B", "Nelder-Mead", "subplex"), control = list(maxit = 20000), precalc = NULL, diagnostic = TRUE, echo = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree</td>
<td>Phylogenetic tree (phylo object).</td>
</tr>
<tr>
<td>data</td>
<td>Matrix or data frame with species in rows and continuous traits in columns (preferentially with names and in the same order than in the tree).</td>
</tr>
<tr>
<td>error</td>
<td>Matrix or data frame with species in rows and continuous trait standard errors (squared) in columns.</td>
</tr>
<tr>
<td>param</td>
<td>List of arguments to be passed to the function. See details.</td>
</tr>
<tr>
<td>method</td>
<td>Choose between &quot;rpf&quot;, &quot;sparse&quot;, &quot;inverse&quot;, &quot;pseudoinverse&quot;, or &quot;pic&quot; for computing the log-likelihood during the fitting process. See details.</td>
</tr>
<tr>
<td>scale.height</td>
<td>Whether the tree should be scaled to length 1 or not.</td>
</tr>
<tr>
<td>optimization</td>
<td>Methods used by the optimization routines. (See ?optim and ?subplex for details).</td>
</tr>
<tr>
<td>control</td>
<td>Max. bound for the number of iteration of the optimizer; other options can be fixed on the list (See ?optim or ?subplex for details).</td>
</tr>
</tbody>
</table>
mvEB

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>precalc</td>
<td>Optional. precalculation of fixed parameters. See ?mvmorph.Precalc for details.</td>
</tr>
<tr>
<td>diagnostic</td>
<td>Whether the diagnostics of convergence should be returned.</td>
</tr>
<tr>
<td>echo</td>
<td>Whether the results must be returned or not.</td>
</tr>
</tbody>
</table>

Details

The Early Burst model (Harmon et al. 2010) is a special case of the ACDC model of Blomberg et al. (2003). Using an upper bound larger than zero transform the EB model to the accelerating rates of character evolution of Blomberg et al. (2003).

The "method" argument allows the user to try different algorithms for computing the log-likelihood. The "rpf" and "sparse" methods use fast GLS algorithms based on factorization for avoiding the computation of the inverse of the variance-covariance matrix and its determinant for the log-likelihood estimation. The "inverse" approach uses the "stable" standard explicit computation of the inverse and determinant of the matrix and is therefore slower. The "pseudoinverse" method uses a generalized inverse that is safer for matrix near singularity but very time consuming. The "pic" method uses a very fast algorithm based on independent contrasts. It should be used with strictly dichotomic trees (i.e., no polytomies) and is currently not available for the multivariate "BMM" model. See ?mvLL for more details on these computational methods.

The "param" list can be used to set the lower (low, default value is -3) and upper (up, default value is 0 - i.e., Early Burst model) bounds for the estimation of the exponential rate (beta). The lower bound for decelerating rates (as assumed in Early Burst) could be fixed as log(min.rate) / T, where T is the depth of the tree and min.rate is the minimum rate that could be assumed for the model (following Slater and Pennell, 2014)

Starting values for "sigma" and "beta" could also be provided through the "param" list.

Value

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogLik</td>
<td>The log-likelihood of the optimal model.</td>
</tr>
<tr>
<td>AIC</td>
<td>Akaike Information Criterion for the optimal model.</td>
</tr>
<tr>
<td>AICC</td>
<td>Sample size-corrected AIC.</td>
</tr>
<tr>
<td>theta</td>
<td>Estimated ancestral states.</td>
</tr>
<tr>
<td>beta</td>
<td>Exponent rate (of decay or increase).</td>
</tr>
<tr>
<td>sigma</td>
<td>Evolutionary rates matrix for each selective regimes.</td>
</tr>
<tr>
<td>convergence</td>
<td>Convergence status of the optimizing function; &quot;0&quot; indicates convergence (See ?optim for details).</td>
</tr>
<tr>
<td>hessian</td>
<td>Hessian matrix of second order partial derivatives at the MLE. (See ?mvOU for details).</td>
</tr>
<tr>
<td>hess.values</td>
<td>Reliability of the likelihood estimates calculated through the eigen-decomposition of the hessian matrix. &quot;0&quot; means that a reliable estimate has been reached. (See ?mvOU for details).</td>
</tr>
<tr>
<td>param</td>
<td>List of model fit parameters (optimization, method, model, number of parameters...).</td>
</tr>
</tbody>
</table>

Author(s)

Julien Clavel
References


See Also

mvMORPH mvOU mvBM mvSHIFT mvSIM optim

Examples

## Toy example

```r
# Simulated dataset
set.seed(14)
# Generating a random tree
tree<-pbtree(n=50)

# Setting the regime states of tip species
sta<-as.vector(c(rep("Forest",20),rep("Savannah",30))); names(sta)<-tree$tip.label

# Making the simmap tree with mapped states
tree<make.simmap(tree,sta, model="ER", nsim=1)
col<"blue","orange"; names(col)<-c("Forest","Savannah")

# Plot of the phylogeny for illustration
plotSimmap(tree,col,fsize=0.6,node.numbers=FALSE,lwd=3, pts=FALSE)

# Simulate the traits
sig1=matrix(c(0.1,0.05,0.05,0.1),2)
theta<0
beta<-0.8
data<mvSIM(tree, param=list(sigma=sigma, beta=beta, ntraits=2, mu=theta,
 names_traits=c("head.size","mouth.size")), model="EB", nsim=1)

## Fitting the models
mvEB(tree, data)
mvEB(tree, data, method="pic")

# ACDC
beta<- 0.5
data<mvSIM(tree, param=list(sigma=sigma, beta=beta, ntraits=2, mu=theta,
 names_traits=c("head.size","mouth.size")), model="EB", nsim=1)

fit<mvEB(tree, data, method="pic", param=list(up=2, low=-2))
```
mvLL

Multivariate (and univariate) algorithms for log-likelihood estimation of arbitrary covariance matrix/trees

Description

This function allows computing the log-likelihood and estimating ancestral state of arbitrary tree or variance-covariance matrix with different algorithms based on GLS (Generalized Least Square) and Independent Contrasts. Works for univariate or multivariate models. Can be wrapped for maximizing the log-likelihood of user-defined models.

Usage

mvLL(tree, data, error = NULL, method = c("pic", "rpf", "sparse", "inverse", "pseudoinverse"), param = list(estim = TRUE, mu = 0, sigma = 0, D = NULL, check = TRUE), precalc = NULL)

Arguments

tree Whether a phylogenetic tree of class "phylo" or a variance-covariance matrix (vcv) of that tree.
data Matrix or data frame with species in rows and continuous traits in columns.
error Matrix or data frame with species in rows and continuous trait standard errors (squared) in columns.
method Method used for computing the log-likelihood. Could be "pic", "sparse", "rpf", "inverse", or "pseudoinverse". See details below.
param List of additional arguments to be passed through the function. The "estim", "mu" and "sigma" arguments are only used with the "pic" method. The "D" argument is used with the others to specify the design matrix. See details below.

Details

The mvLL function computes the log-likelihood and the ancestral states (mean at the root-theta) for arbitrary variance-covariance matrix (or trees for the pruning algorithm based on independent contrasts "pic") provided by the user. This function can be wrapped for optimizing various multivariate models of trait evolution (by transforming the branch lengths of a tree for the "pic" method, or feeding it with variance-covariance and design matrices for the others methods).

Five methods are proposed to compute the log-likelihood:
"pic" is a very fast pruning algorithm based on independent contrasts which should be used with strictly dichotomic trees (i.e., no polytomies). This method can neither be used with measurement errors nor for multiple ancestral states estimation (theta values).

"rpf" is a GLS algorithm using the rectangular packed format Cholesky factorization for solving the linear system without computing the inverse of the variance-covariance matrix and its determinant (normally used in the loglikelihood estimation). This algorithm uses fast BLAS 3 routines with half storage in packed format for computing the Cholesky upper factor. This method is more efficient than the "inverse" method and can be used with dense matrices (no zero entries).

"sparse" is a GLS algorithm using Cholesky factorization for sparse matrices (including zero entries). The matrices are stored in the "old Yale sparse format" internally. Depending on the sparsity structure of the variance-covariance matrix this algorithm can be more efficient than the "rpf" method.

"inverse" is a GLS algorithm that uses explicit inversion of the variance-covariance matrix (through QR decomposition) as well as computation of its determinant in the log-likelihood estimation. This is the "textbook" method, that is computationally more intensive than the previous approaches.

"pseudoinverse" is a GLS method that uses a generalized inverse (through SVD) for computing the log-likelihood. This method is safer when the matrix is near singularity, but it is the most time-consuming.

The user must provide a variance-covariance matrix (e.g., vcv.phylo(tree)) or a multivariate variance-covariance matrix (e.g., kronecker(matrix(c(2,1,1,1.5),2),vcv.phylo(tree)) as well as a design matrix (or multivariate design matrix) with the "rpf", "sparse", "inverse", and "pseudoinverse" methods.

Use the "param" list of arguments to define whether or not the brownian rate should be estimated and returned (estim=TRUE) with the "pic" method. Otherwise, the rate parameter (also called sigma) is fixed to 1. The arguments "mu" and "sigma" can be used to specify (e.g., in a MCMC setting) the mean at the root and the brownian rate, respectively.

You can choose to provide differently scaled trees for multivariate data with the "pic" method. In such a case, the trees (one per trait) should be embedded within a list() object. See example below.

Value

logl  Estimated log-likelihood for the data with the given matrix/tree.
theta Estimated ancestral states at the root. They are defined by the design matrix (D) for all the methods but "pic".
sigma  Estimated rate(s) parameters. Only when param$estim=TRUE with the "pic" method.

Author(s)

Julien Clavel

References


See Also

mvMORPH mvOU mvEB mvBM mvSHIFT mvSIM

Examples

```r
## Simulated dataset
set.seed(14)
# Generating a random tree with 50 tips
n=50
tree<-pbtree(n=n)

# Simulated trait
data=rTraitCont(tree)

# Design matrix
d=matrix(rep(1,n),ncol=1)

## Compute the log-likelihood
## Inverse
mvLL(ccv.phylo(tree),data,method="inverse",param=list(D=D))

## Pseudoinverse
mvLL(ccv.phylo(tree),data,method="pseudoinverse",param=list(D=D))

## Sparse
mvLL(ccv.phylo(tree),data,method="sparse",param=list(D=D))

## RPF
mvLL(ccv.phylo(tree),data,method="rpf",param=list(D=D))

## Pic
mvLL(tree,data,method="pic",param=list(estim=TRUE))

## Pic with arbitrary values
mvLL(tree,data,method="pic",param=list(estim=FALSE, mu=0, sigma=1))
mvLL(tree,data,method="pic",param=list(estim=FALSE))
mvLL(tree,data,method="pic",param=list(estim=FALSE, sigma=1))

## Multivariate cases
## Simulate traits
data2<mvSIM(tree,nsim=1,model="BM1",param=list(sigma=diag(2),mu=c(0,0),ntraits=2))
# Design matrix
d<-cbind(rep(c(1,0),each=50),rep(c(0,1),each=50))

## RPF
mvLL(kronecker(diag(2),ccv.phylo(tree)),data2,method="rpf", param=list(D=D))

## Inverse (with default design matrix if not provided)
```
mvLL(kronecker(diag(2), vcv.phylo(tree)), data2, method = "inverse")

# Pic
mvLL(tree, data2, method = "pic")
# NB: The trees in the list could be differently scaled for each traits...
mvLL(list(tree, tree), data2, method = "pic")

## VERY FAST COMPUTATION FOR LARGE TREES (not run, but take few seconds)
# Big tree (1,000,000 species)
# tree2 <- rtree(1000000)
# Simulate trait with a Brownian motion process
# trait <- rTraitCont(tree2)
# system.time(mvLL(tree2, trait, method = "pic", param = list(estim = FALSE, sigma = 1)))
#
# precalc <- mv.Precalc(tree2, nb.traits = 1, param = list(method = "pic"))
# system.time(mvLL(tree2, trait, method = "pic", param = list(estim = FALSE, sigma = 1),
#           precalc = precalc))
#
# Check = FALSE !! Your tree should be in post-order !!
# tr2 <- reorder(tree2, "postorder")
# system.time(mvLL(tr2, trait, method = "pic", param = list(estim = FALSE, sigma = 1, check = FALSE)))

---

mvOU  

**Multivariate Ornstein-Uhlenbeck model of continuous traits evolution**

**Description**

This function allows the fitting of a multivariate Ornstein-Uhlenbeck (OU) model by allowing a given tree branch to be subdivided into multiple selective regimes using SIMMAP-like mapping of ancestral states. Species measurement errors or dispersions can also be included in the model.

**Usage**

```r
mvOU(tree, data, error = NULL, model = c("OU", "OU1"), param = list(sigma = NULL, alpha = NULL, vcv = "mvn Morph", decomp = c("symmetricPositive", "symmetric", "nsymPositive", "nsymmetric", "diagonal"), method = c("rpf", "sparse", "inverse", "pseudoinverse", "univarpf"), scale.height = FALSE, optimization = c("L-BFGS-B", "Nelder-Mead", "subplex"), control = list(maxit = 20000), precalc = NULL, diagnostic = TRUE, echo = TRUE)
```

**Arguments**

- **tree**: Phylogenetic tree with mapped ancestral states in SIMMAP format. (See make.simmap function from phytools package). A "phylo" object can be used with model "OU1".
- **data**: Matrix or data frame with species in rows and continuous traits in columns.
- **error**: Matrix or data frame with species in rows and continuous trait standard errors (squared) in columns.
The `mvOU` function fits a multivariate model of evolution according to an Ornstein-Uhlenbeck process. The user can incorporate measurement errors and uses SIMMAP-like mapping of ancestral states. SIMMAP mapping allows one to assign parts of branches to different selective regimes, and allows testing for change in trait variance that is not synchronous with the species divergence events.

Mapping of ancestral states can be done using the "make.simmap", "make.era.map" or "paintSubTree" functions from the "phytools" package.

The "method" argument allows the user to try different algorithms for computing the log-likelihood. The "rpf", "univaripf" (for univariate analysis) and "sparse" methods use fast GLS algorithms based on factorization for avoiding the computation of the inverse of the variance-covariance matrix and its determinant for the log-likelihood estimation. The "inverse" approach uses the "stable" standard explicit computation of the inverse and determinant of the matrix and is therefore slower. The "pseudoinverse" method uses a generalized inverse that is safer for matrix near singularity but very time consuming. See ?mvLL for details.

Arguments in the "param" list are:

"sigma" or "alpha" - Starting values for the likelihood search can be specified through the "alpha" and "sigma" arguments in the param list. It is also possible to test for the significance of the off-diagonal sigma and alpha matrix in the full model by making comparison with a constrained model (using sigma="constraint", or alpha="constraint") in the "param" argument list. You can also provide starting values for the constrained model. For instance, for two traits use sigma=list("constraint", c(0.5,0.5)) (or alpha=list("constraint", c(0.5,0.5))).

"decomp" - You can further constraint the alpha matrix by specifying the decomposition of the matrix through the "decomp" argument in the "param" list. Indeed, the multivariate Ornstein-Uhlenbeck model is described by the spectral decomposition of the alpha matrix. Thus it is possible to parameterize the alpha matrix on its eigenvalues with different biological interpretations (e.g., Sy et al. 1997, Bartoszek et al. 2012). "symmetric" uses SVD parameterization of the matrix, "symmetricPositive" uses a Cholesky parameterization of the matrix, "nsymmetric" and "nsymPositive" use a Schur parameterization of the matrix for specifying the eigenvalues. It is also possible to specify "diagonal" which is similar to the use of the "constraint" argument for "alpha" argument.
The sigma matrix is parameterized by the Cholesky method to ensure positive definiteness (Pinheiro and Bates, 1996).

"vcv" - It is possible to specify in the "param" list what kind of variance-covariance matrix to use with the "vcv" argument. The vcv="ouch" option uses a modified version of the OUCH package code (Butler and King, 2004) to compute the variance-covariance matrix. This algorithm assumes that the process is stationary. It cannot be used with the "sparse" method to speed up the computations. The vcv="mvmorph" option follows Bartoszek et al. 2012 for computing the variance covariance matrix. This approach can be used with non-ultrametric trees, with the various matrix parameterization proposed and all the GLS methods of mvMORPH. Both approaches should converge on the same results when the traits have evolved according to an OU process.

"root" - This argument allows the user to specify if the ancestral state at the root (theta 0) should be estimated (root=TRUE), assumed to be at the oldest regime state (root=FALSE), or that there is no root and each regime is at the stationary point (root="stationary"). The first option should be used with non-ultrametric trees (i.e., with fossil species - e.g., Hansen 1997) where information on the ancestral state is directly available from the data. Indeed, estimating shifts in the ancestral state from extant species could be problematic and it seems preferable to either assume each regime to be stationary or that the root state is the state of the older regime (as currently assumed in the "ouch" package).

Note on the returned Hessian matrix in the result list:

The hessian is the matrix of second order partial derivatives of the likelihood function with respect to the maximum likelihood parameter values. This matrix provides a measure of the steepness of the likelihood surface in the region of the optimum. The eigen-decomposition of the hessian matrix returned by the optimizing function allows assessing the reliability of the fit of the model (even if the optimizer have converged). When the optimization function does not converge on a stable result, the user may consider increasing the "maxit" argument in the "control" option, or try a simpler model with fewer parameters to estimate. Changing the starting values ("alpha" and "sigma" options in the param list) as well as the optimizing function ("optimization" options) may help sometimes (e.g., alpha=runif(3) for two traits analysis with random starting values (i.e., the lower triangular alpha matrix)). Note that the number of starting values to provide depends on the matrix decomposition chosen for the alpha parameter (p*(p+1)/2 values for symmetric alpha matrix but p*p values for non-symmetric ones - with p the number of traits).

Note that Bartoszek et al. (2012) proposed the mvSLOUCH package dedicated to multivariate Ornstein-Uhlenbeck processes, which allows fitting models with a randomly evolving predictor variables. They also provide a detailed mathematical description of the multivariate OU model which the reader should consult.

Value

| LogLik     | The log-likelihood of the optimal model. |
| AIC        | Akaike Information Criterion for the optimal model. |
| AICc       | Sample size-corrected AIC. |
| theta      | Estimated ancestral states. |
| alpha      | Matrix of estimated alpha values (strength of selection). |
| sigma      | Evolutionary rates matrix (drift). |
| convergence| Convergence status of the optimizing function; "0" indicates convergence. (See ?optim for details). |
hessian

Hessian matrix of second order partial derivatives at the MLE. (See details).

hess.values

Reliability of the likelihood estimates calculated through the eigen-decomposition of the hessian matrix. "0" means that a reliable estimate has been reached. See details.

param

List of model fit parameters (optimization, method, model, number of parameters...).

Note

This function partly uses a modified version of the C code from the "OUCH" package built by Aaron King, as well as a C code which is part of the "ape" package by Emmanuel Paradis. I kindly thank those authors for sharing their sources.

Author(s)

Julien Clavel

References


See Also

mvMORPH mvBM mvEB mvSHIFT mvSIM LRT optim make.simmap make.era.map paintSubTree

Examples

```r
## Toy example

# Simulated dataset
set.seed(14)
# Generating a random tree
tree<-pbtree(n=50)

# Setting the regime states of tip species
sta<-as.vector(c(rep("Forest",20),rep("Savannah",30))); names(sta)<-tree$tip.label
```
# Making the simmap tree with mapped states
```
tree<--make.simmap(tree, sta, model="ER", nsim=1)
col<--c("blue","orange"); names(col)<--c("Forest","Savannah")
```

# Plot of the phylogeny for illustration
```
plotSimmap(tree, col, fsize=0.6, node.numbers=FALSE, lwd=3, pts=FALSE)
```

# Simulate the traits
```
alpha<--matrix(c(2,0.5,0.5,1,2)
sigma<--matrix(c(0.1,0.05,0.05,0.1),2)
theta<--c(2,3,1,1.3)
data<--mvSIM(tree, param=list(sigma=sigma, alpha=alpha, ntraits=2, mu=theta, names_traits=c("head.size","mouth.size")), model="OUM", nsim=1)
```

## Fitting the models
### OUM - Analysis with multiple optima
```
mvOU(tree, data)
```

### OU1 - Analysis with a unique optimum
```
mvOU(tree, data, model="OU1", method="sparse")
```

### Various options
```
mvOU(tree, data, model="OUM", method="sparse", scale.height=FALSE, param=list(decomp="nsymmetric", root="stationary"))
mvOU(tree, data, model="OUM", method="sparse", scale.height=FALSE, param=list(decomp="nsymmetric", root=TRUE))
mvOU(tree, data, model="OUM", method="sparse", scale.height=FALSE, param=list(decomp="symmetricPositive", root=TRUE))
```

### OUCH setting
```
mvOU(tree, data, model="OUM", method="rpf", scale.height=FALSE, param=list(decomp="symmetricPositive", root=FALSE, vcv="ouch"))
```

### Univariate case - FAST with RPF
```
set.seed(14)
tree<--pbtree(n=500)
```

### Setting the regime states of tip species
```
sta<--as.vector(c(rep("Forest",200),rep("Savannah",300))); names(sta)<--tree$tip.label
```

# Making the simmap tree with mapped states
```
tree<--make.simmap(tree, sta, model="ER", nsim=1)
col<--c("blue","orange"); names(col)<--c("Forest","Savannah")
```

# Plot of the phylogeny for illustration
```
plotSimmap(tree, col, fsize=0.6, node.numbers=FALSE, lwd=3, pts=FALSE)
```

### Parameters
```
alpha<--2.5
sigma<--0.1
theta<--c(0,2)
data<--mvSIM(tree, param=list(sigma=sigma, alpha=alpha, ntraits=1, mu=theta, names_traits=c("body.size")), model="OUM", nsim=1)
```
mvSHIFT

Multivariate change in mode of continuous traits evolution

Description

This function fits different models of evolution after a fixed point. This allows fitting models of change in mode of evolution following a given event.

Usage

mvSHIFT(tree, data, error = NULL, param = list(age = NULL, sigma = NULL, alpha = NULL, sig = NULL, beta = NULL), model = c("ER", "RR", "EC", "RC", "SR", "EBOU", "OUEB", "EBBM", "BMEB"), method = c("rpf", "sparse", "inverse", "pseudoinverse"), scale.height = FALSE, optimization = c("L-BFGS-B", "Nelder-Mead", "subplex"), control = list(maxit = 20000), precalc = NULL, diagnostic = TRUE, echo = TRUE)

Arguments

tree Phylogenetic tree with shift mapped. (See "make.era.map" function from "phytools" package). A "phylo" object can be used if the "age" argument is provided in the "param" list.
data Matrix or data frame with species in rows and continuous traits in columns.
error Matrix or data frame with species in rows and continuous trait standard errors (squared) in columns.
param List of arguments to be passed to the function. See details.
model Choose between the different models "OUBM", "BMOU", "EBOU", "OUEB", "BMEB", "EBBM"... See details below.
method Choose between "rpf", "sparse", "inverse", or "pseudoinverse" for computing the log-likelihood during the fitting process. See details.
scale.height Whether the tree should be scaled to length 1 or not.
optimization Methods used by the optimization routines. (See ?optim and ?subplex for details).
The `mvSHIFT` function fits a shift in mode or rate of evolution at a fixed point in time, as previously proposed by some authors (O’Meara et al. 2006; O’Meara, 2012; Slater, 2013). Shift in mode of evolution could be mapped on a modified 'phylo' object using the 'make.era.map' function from the 'phytools' package. Note that only one shift is allowed by the current version. The age of the shift could be otherwise directly provided (in unit of times of the tree) in the function by the "age" argument in the "param" list.

The function allows fitting model with shift from an Orstein-Uhlenbeck to a Brownian motion process and vice-versa ("OUBM" and "BMOU"), shifts from a Brownian motion to/from an Early Burst (ACDC) model ("BMEB" and "EBBM"), or shifts from an Orstein-Uhlenbeck to/from an Early Burst (ACDC) model ("OUEB" and "EBOU").

In all these cases it is possible to allow the drift parameter to vary after the fixed point by specifying "i" (for independent) after the model name. For instance, to fit models of "ecological release" or "ecological release and radiate" following Slater (2013), we can use "OUBM" or "OUBMi" respectively.

Alternatively it is also possible to use the shortcuts "ER" or "RR" to fit respectively models of "ecological release" and "ecological release and radiate", and "EC" for a model of "constrained ecology" (e.g., after invasion of a competitive species in a given ecosystem) where traits are constrained in an Ornstein-Uhlenbeck process after a fixed point in time ("RC" is the same model but assumes an independent rate during the early radiative phase). The "SR" model allows fitting different (Brownian) rates/drift before and after the shift point (note that this model could also be fitted using the `mvBM` function).

The "param" list can be used to provide lower and upper bounds for the exponential rate parameter of the Early-Burst/ACDC model. See `?mvEB` for details.

The "method" argument allows the user to try different algorithms for computing the log-likelihood. The "rpf" and "sparse" methods use fast GLS algorithms based on factorization for avoiding the computation of the inverse of the variance-covariance matrix and its determinant involved in the log-likelihood estimation. The "inverse" approach uses the "stable" standard explicit computation of the inverse and determinant of the matrix and is therefore slower. The "pseudoinverse" method uses a generalized inverse that is safer for matrix near singularity but very time consuming. See `?mvLL` for details.

**Value**

- **LogLik**: The log-likelihood of the optimal model.
- **AIC**: Akaike Information Criterion for the optimal model.
- **AICc**: Sample size-corrected AIC.
- **theta**: Estimated ancestral states.
alpha  Matrix of estimated alpha values (strength of selection).
beta   Exponent rate (of decay or increase) for the ACDC/Early-Burst model.
sigma  Evolutionary rates matrix (drift) for the BM process before the shift.
sig    Evolutionary rates matrix (drift) for the BM process after the shift (only for "i" models).
convergence  Convergence status of the optimizing function; "0" indicates convergence. (See ?optim for details).
hessian Hessian matrix of second order partial derivatives at the MLE. (See ?mvOU for details).
hess.values Reliability of the likelihood estimates calculated through the eigen-decomposition of the hessian matrix. "0" means that a reliable estimate has been reached. (See ?mvOU for details).
param List of model fit parameters (optimization, method, model, number of parameters...).

Note
Changes in rate of evolution and optima can also be fitted using the mvBM and mvOU functions using a 'make.era.map' transformed tree.

Author(s)
Julien Clavel

References

See Also
mvMORPH mvOU mvBM mvEB mvSIM optim subplex paintSubTree make.era.map

Examples

```r
## Toy example

# Simulated dataset
set.seed(14)
# Generating a random tree
tree<-rtree(50)

# Providing a tree with the shift mapped on
```
mvSHIFT

tot<-max(nodeHeights(tree))
age=tot-3  # The shift occurred 3 Ma ago

tree<-make.era.map(tree,c(0,age))

# Plot of the phylogeny for illustration
plotSimmap(tree,fsize=0.6,node.numbers=FALSE,lwd=3, pts=FALSE)

# Simulate the traits
alpha<-.matrix(c(2,0.5,0.5,1,2)
sigma<-.matrix(c(0.1,0.05,0.05,0.1,2)
theta<-c(2,3)
data<-mvSIM(tree, param=list(sigma=sigma, alpha=alpha, ntraits=2, mu=theta, names_traits=c("head.size","mouth.size")), model="ER", nsim=1)

## Fitting the models
# "Ecological release model"
mvSHIFT(tree, data, model="OUBM") # similar to mvSHIFT(tree, data, model="ER")

# "Release and radiate model"
# mvSHIFT(tree, data, model="RR", method="sparse")
# similar to mvSHIFT(tree, data, model="OUBM")

# More generally...
# OU to/from BM
# mvSHIFT(tree, data, model="OUBM", method="sparse")
# mvSHIFT(tree, data, model="BMOU", method="sparse")
# mvSHIFT(tree, data, model="OUBMi", method="sparse")
# mvSHIFT(tree, data, model="BMOUi", method="sparse")

# BM to/from EB
# mvSHIFT(tree, data, model="BMEB", method="sparse")
# mvSHIFT(tree, data, model="EBBM", method="sparse")
# mvSHIFT(tree, data, model="BMEBi", method="sparse")
# mvSHIFT(tree, data, model="EBBMi", method="sparse")

# OU to/from EB
# mvSHIFT(tree, data, model="OUEB", method="sparse")
# mvSHIFT(tree, data, model="OUEBi", method="sparse")
# mvSHIFT(tree, data, model="EOBU", method="sparse")
# mvSHIFT(tree, data, model="EOBUi", method="sparse")

## Without providing mapped tree
# The shift occurred 3Ma ago (param$age=3)
# set.seed(14)
# tree<-rtree(50)
# data<-mvSIM(tree, param=list(sigma=sigma, alpha=alpha, ntraits=2, mu=theta, names_traits=c("head.size","mouth.size"), model="ER", age=3), nsim=1)

## Fitting the models without mapped tree but by specifying the age in the param list.
# mvSHIFT(tree, data, model="OUBM", param=list(age=3))
mvSIM

Simulation of (multivariate) continuous traits on a phylogeny

Description

This function allows simulating multivariate (as well as univariate) continuous traits evolving according to a BM (Brownian motion), OU (Ornstein-Uhlenbeck), ACDC (Accelerating rates and Decelerating rates/Early bursts), and SHIFT models of phenotypic evolution.

Usage

```r
mvSIM(tree, nsim = 1, error = NULL, model = c("BM", "BMM", "OU", "OUM", "EB"),
      param = list(mu = 0, sigma = 0.1, alpha = 1, beta = 0))
```

Arguments

- `tree`: Phylogenetic tree with mapped ancestral states in SIMMAP format. (See make.simmap function from phytools package) or a standard "phylo" object (ape).
- `nsim`: The number of simulated traits (or datasets for multivariate analysis).
- `error`: Matrix or data frame with species in rows and continuous trait standard errors (squared) in columns.
- `model`: The model of trait evolution for the simulations. Could be any of the models used by the mvBM, mvEB, mvOU and mvSHIFT functions.
- `param`: List of parameter arguments used for the simulations. You should provide the sigma (values or matrix), alpha (for OU and SHIFT models), beta (EB and SHIFT), theta (ancestral states), ntraits (the number of traits) or others param arguments used in the models. Alternatively you can provide a fitted object of class "mvmorph". See details.

Details

This function simulates multivariate (as well as univariate) continuous traits evolving according to a BM (Brownian motion), OU (Ornstein-Uhlenbeck), ACDC (Accelerating rates and Decelerating rates/Early Bursts), and SHIFT models of phenotypic evolution. The traits are simulated by random sampling from a Multivariate Normal Distribution (Paradis, 2012). Phylogenetic tree with mapped ancestral states in SIMMAP format (see make.simmap function from phytools package) or a standard "phylo" object (ape).

The mvSIM function allows simulating continuous trait (univariate or multivariate) evolution along a phylogeny with user specified parameters or estimated parameters from a previous fit.

The "simulate" wrapper can also be used with a fitted object of class "mvmorph": simulate(object, nsim=1, tree=tree). See example.

If parameters values are not provided, the default values are fixed to 1 (sigma, sig, alpha, beta) or to 0 for the mean at the root (ancestral state).
For the "BMM" model were different parts of the tree have their own rate, a list with one rate (or matrix of rates) per selective regimes must be provided.

For the "OU1" and "OUM" models, the user can define if the ancestral state (theta0) should be computed (param$root=TRUE), assumed to be at the oldest regime state (param$root=FALSE), or that there is no root and each regimes is at the stationary point (param$root="stationary"). (See also ?mvOU).

Traits names can be provided with the "names_traits" argument in the "param" list. For all the shift models, if the tree is not mapped the age of the shift should be directly provided (in unit of times of the tree) by the "age" argument in the "param" list.

Value

A matrix with simulated traits (columns) for the univariate case, or a list of matrix for the multivariate case (with nsim>1).

Note

Ancestral states for Ornstein-Uhlenbeck processes (param$root=TRUE) should be used with non-ultrametric trees. As this method uses Multivariate Normal distribution (MVN) for simulating the traits, it is advised to avoid its use with very large datasets/trees and rely instead on recursive algorithms (see e.g., ?rTraitCont from "ape").

Author(s)

Julien Clavel

References


See Also

mvMORPH mvOU mvEB mvBM mvSHIFT mvLL

Examples

```r
## Simulated dataset
set.seed(14)
# Generating a random tree with 50 species
tree<-pbtree(n=50)

# Setting the regime states of tip species
sta<-as.vector(c(rep("Forest",20),rep("Savannah",30))); names(sta)<-tree$tip.label

# Making the simmap tree with mapped states
tree<-make.simmap(tree,sta , model="ER", nsim=1)
col<-c("blue","orange"); names(col)<-c("Forest","Savannah")

# Plot of the phylogeny for illustration
```
mvSIM

```r
plotSimmap(tree,col,fs=0.6,node.numbers=FALSE,lwd=3, pts=FALSE)

## Simulate trait evolution according to a bivariate "BMM" model
# Number of traits
ntraits<-2
# Number of simulated (pairs of) traits
nsim<-10
# Rates matrices for the "Forest" and the "Savannah" regimes
sigma<-list(Forest=matrix(c(2,0.5,0.5,1,2), Savannah=matrix(c(5,3,3,4,2)))
# ancestral states for each traits
theta<-c(0,0)

# Simulate
simul<-mvSIM(tree,nsim=nsim, model="BMM", param=list(ntraits=ntraits,sigma=sigma, mu=theta))

# Try to fit a "BM1" model to the first simulated dataset
model_fit<-mvBM(tree,simul[[1]],model="BM1")

# Use the estimated parameters to simulate new traits!
simul2<-mvSIM(tree,nsim=nsim,param=model_fit)

# or try with generic "simulate" function
simul3<-simulate(model_fit,nsim=nsim,tree=tree)

## Just-for-fun :comparing parameters
simul4<-simulate(model_fit,nsim=100,tree=tree)
results<-lapply(1:100,function(x){
  # mvBM(tree,simul4[[x]],model="BM1",method="pic", echo=F,diagnostic=F)
  # })
  
  # sigma_simul<-sapply(1:100,function(x)(results[[x]]$sigma))
  # comparison between the simulated (black) and the observed (red) multivariate rates
  layout(matrix(1:4, ncol=2))
  for(i in 1:4){
    hist(sigma_simul[i,], main=paste("Estimated sigma on simulated traits"),
    xlab="estimated sigma for 100 replicates");abline(v=mean(sigma_simul[i,]),lwd=2); 
    abline(v=model_fit$sigma[i],col="red",lwd=2)
  }
  
```
## Index

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>BM</td>
<td>2</td>
</tr>
<tr>
<td>Brownian Motion</td>
<td>6, 21</td>
</tr>
<tr>
<td>Cholesky constraint</td>
<td>6</td>
</tr>
<tr>
<td>EB</td>
<td>2</td>
</tr>
<tr>
<td>EC</td>
<td>21</td>
</tr>
<tr>
<td>ER</td>
<td>21</td>
</tr>
<tr>
<td>Early-Burst</td>
<td>21</td>
</tr>
<tr>
<td>Evolutionary rates</td>
<td>6, 2, 21</td>
</tr>
<tr>
<td>GLS</td>
<td>13</td>
</tr>
<tr>
<td>Hessian</td>
<td>16</td>
</tr>
<tr>
<td>Independent contrasts</td>
<td>13</td>
</tr>
<tr>
<td>LRT</td>
<td>2</td>
</tr>
<tr>
<td>Loglikelihood ratio test</td>
<td>2</td>
</tr>
<tr>
<td>Loglikelihood</td>
<td>13</td>
</tr>
<tr>
<td>Measurement error</td>
<td>2</td>
</tr>
<tr>
<td>Methods</td>
<td>13</td>
</tr>
<tr>
<td>Models comparison</td>
<td>2</td>
</tr>
<tr>
<td>OU</td>
<td>2</td>
</tr>
<tr>
<td>Ornstein Uhlenbeck</td>
<td>16, 6, 21</td>
</tr>
<tr>
<td>RR</td>
<td>21</td>
</tr>
<tr>
<td>SIMMAP</td>
<td>2</td>
</tr>
<tr>
<td>Shifts</td>
<td>2</td>
</tr>
<tr>
<td>Simulations</td>
<td>2</td>
</tr>
<tr>
<td>\textasciitilde kwd1</td>
<td>mv.Precalc, 4, mvEB, 10, mvSIM, 25</td>
</tr>
<tr>
<td>\textasciitilde kwd2</td>
<td>mv.Precalc, 4, mvEB, 10, mvSIM, 25</td>
</tr>
<tr>
<td>brownie.lite</td>
<td>2, 8</td>
</tr>
<tr>
<td>evol.vcv</td>
<td>2, 8</td>
</tr>
<tr>
<td>LRT</td>
<td>2, 2, 8, 19</td>
</tr>
<tr>
<td>make.era.map</td>
<td>8, 19, 23</td>
</tr>
<tr>
<td>make.simmap</td>
<td>8, 19</td>
</tr>
<tr>
<td>mv.Precalc</td>
<td>4</td>
</tr>
<tr>
<td>mvBM</td>
<td>2, 3, 5, 6, 12, 15, 19, 23, 26</td>
</tr>
<tr>
<td>mvEB</td>
<td>2, 3, 5, 8, 10, 15, 19, 23, 26</td>
</tr>
<tr>
<td>mvLL</td>
<td>2, 5, 13, 26</td>
</tr>
<tr>
<td>mvMORPH</td>
<td>3, 5, 8, 12, 15, 19, 23, 26</td>
</tr>
<tr>
<td>mvMORPH (mvMORPH-package)</td>
<td>2</td>
</tr>
<tr>
<td>mvMORPH-package</td>
<td>2</td>
</tr>
<tr>
<td>mvOU</td>
<td>2, 3, 5, 8, 12, 15, 16, 23, 26</td>
</tr>
</tbody>
</table>
INDEX

mvSHIFT, 2, 3, 5, 8, 12, 15, 19, 21, 26
mvSIM, 2, 8, 12, 15, 19, 23, 25
optim, 8, 12, 19, 23
paintSubTree, 8, 19, 23
subplex, 23